View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Periodica Polytechnica (Budapest University of Technology and Economics)

Periodica Polytechnica
Transportation Engineering

44(1), pp. 13-22, 2016
DOI: 10.3311/PPtr.8035
Creative Commons Attribution

RESEARCH ARTICLE

Development of an Intelligent Path Planning Method for Materials Handling Machinery at Construction Sites

Gábor Bohács[1], András Gyimesi[1], Zoltán Rózsa[1*]

Received 05 March 2015; accepted 09 May 2015

Abstract

Construction processes can be implemented in most cases using various sorts of equipment. These can move along various paths, over the construction site from one job to another or during their operation. The proper choice of their path is a basic aspect for the planning of optimal processes. This paper is devoted to this problem. First it surveys some useful methods of path planning from the literature. Then it compares the methods considering construction specialties and finally it presents the developed intelligent system.

Keywords

Path planning, Construction logistics, Fuzzy

[1] Department of Material Handling and Logistics Systems,
Faculty of Transportation Engineering and Vehicle Engineering,
Budapest University of Technology and Economics,
Bertalan L. u. 7., H-1111 Budapest, Hungary

Gábor Bohács Researcher ID: F-4539-2015
András Gyimesi Researcher ID: F-4819-2015
Zoltán Rózsa Researcher ID: F-4819-2015

[*] Corresponding author, e-mail: zoltan.rozsa@logisztika.bme.hu

1 Introduction

Construction is one of the most important industries in the world. It creates valuable objects using expensive machinery. As building processes are complex, each project can be regarded as a unique one. Layout planning is an important issue in the planning phase of the construction. After the creation of the optimal layout, path planning can be implemented. The transport of materials often relates to logistics (Andrejiova et al., 2014). Using planning algorithms for optimized paths, its positive effect is significant in terms of time saving, cost, and safety. This optimization is mainly a heuristic task, which requires expert knowledge. In this paper, surveying applied methods found in the literature was done first, then selecting and improving the most applicable one as well.

2 Study of path planning methods

In this chapter, an overview of previously acknowledged theories and methods found in the literature is presented. First, theories are summarized that come from the origins of path planning, i.e. mobile robotics. In the next chapter, we investigate the construction specific aspects, and finally the applicability of these methods. In case of robotics or construction machines, we can distinguish path planning and Coverage Path Planning (CPP). Path planning is to provide a route for the equipment on the safest and/or shortest way (or the way requiring minimum control effort) while moving from one location to another. These travel activities can be loaded and unloaded runs of transport vehicles, or location change for working machinery. Coverage path planning designs the route in a way, where the equipment reaches more than two points. In general, this is the case during the working phase of the equipment. A great study has already been made by Galceran and Carreras (2013) about the CPP methods. The subject of this paper is related to mainly ordinary path planning. Within this topic, the following models describing the environment are distinguished:

- Static environment with complete information,
- Static environment with incomplete information,
- Predictable (complete information), dynamic environment,

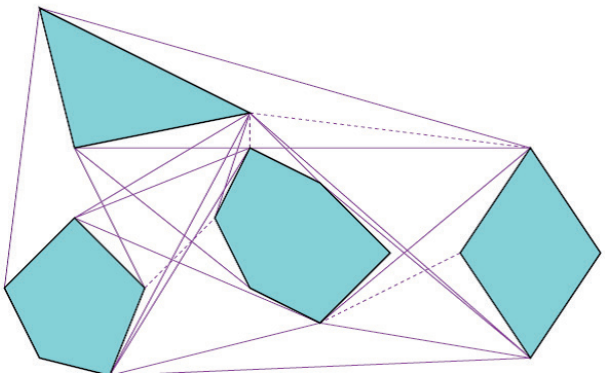

Fig. 1 Visibility graph of convex polygons (van den Berg, 2007)

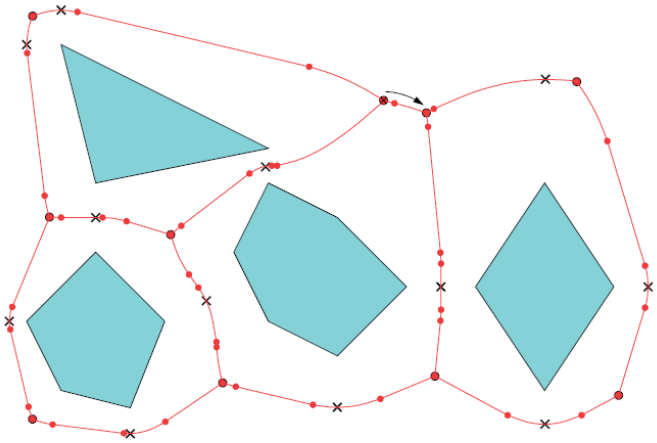

Fig. 2 Voronoi diagram of convex polygons (van den Berg, 2007)

- Unpredictable (incomplete information), dynamic environment. (van den Berg, 2007)

The static environment does not contain moving objects except for the equipment, while there are other moving objects in a dynamic environment, for example, machines and persons as well. (Sharma, 2013; Stehlíková et al., 2014). We can classify models applied in motion planning of construction sites as planning with incomplete (sensors are required for exploring unknown terrain) or planning with complete information (this is typically NP-complete). In case of incomplete information models, the path is computed during the motion by the feedback signal of the sensors. A good description can be found about this kind of pathfinding and the applicable algorithms in (Kunigahalli et al., 1994). It is noted that the formation of the above problem is analogous to the path planning problem in mechatronics regarding the path building of the robot limb.

Path planning algorithms can be global or local ones. Global path planning methods require complete information. Local path planning, on the other hand, is one of the most important fields of sensors and robotics that concentrate on detecting unknown obstacles and terrain, and it also focuses on redesigning the path in response to explored environment or environmental changes. (It is almost impossible to find the optimal global path with this approach).

2.1 Network representations

Before starting the path finding, we need to select the data structure with which we want to represent our map (the construction site) as it determines the navigation method and the applicable path finding algorithms.

One option is the skeletonized structure, (such as the visibility graph (Fig. 1)), which causes restricted traversability, but the algorithms' runtime can be short. If we substitute the obstacles with polygonal objects, their edges will be the vertices of our visibility graph, and the connecting straight lines (graph edges) symbolize the traversable roads. Unfortunately, these roads will be close to the obstacles. Other possibilities for skeletonization are the roadmap and the Voronoi diagram, which are commonly mentioned in the literature also (Yu et al., 2013; van den Berg, 2007). The Voronoi diagram (Fig. 2) is based on seeds (obstacles). The diagram is divided into regions around each seed, where the points inside a region around the corresponding seed are the closest to it. The borders of the regions are the Voronoi edges, which are halfway between two seeds.

Another type of representation is the cellular decomposition, whose advantage over the skeletonized structure is the possibility of representing the obstacles in their real shape. Therefore, calculations will be more precise, and the number of traversable roads is not limited. However, its drawback is that algorithms run slow. In the first step, a grid map has to be laid down, and the cells covering an obstacle have to be avoided in the path finding step. The accuracy of the calculation depends on the size and shape of the cells (consider mesh refinement around the obstacles). For the determination of the optimal values, many methods can be used, e.g. quadtree decomposition. (Burlet et al., 2004)

2.2 Routing algorithms

First, the algorithms connected to the shortest path problem are surveyed. An optimization algorithm was developed by Dijkstra (Dijkstra, 1959), which is used to find the shortest path between two points of a controlled or an uncontrolled graph. This method was modified by Solka (1995), who complemented the algorithm in a way that the shortest path can be found with turns smaller than 45°. Others tried to speed up the algorithm, including (Anastopoulos et al., 2009) with concurrent computing. Lots of other researchers have applied the Dijkstra algorithm in different disciplines as well.

There is another algorithm from graph theory, called Bellman-Ford (Bellman, 1958), which is very similar to that of Djikstra's, solving single source shortest path problem. One of the benefits of this algorithm is that it finds the shortest path even on graphs with negative edge weights. Being of higher order is the reason for its less frequent use. In chronological

order, the next algorithm was the so-called Floyd-Warshall, which finds the shortest lengths between all the pairs of vertices. Later, Donald B. Johnson in (Johnson, 1977) showed the Johnson's algorithm, which is used for sparse graph because it can be faster than the Floyd-Warshall. At construction sites, negative edges need not be considered, and knowing the shortest paths between all pairs of vertices is not necessary either.

Another path finding algorithm, called A* was published by Peter E. Hart (Hart et al., 1968) and corrected in (Hart et al., 1972). It adds G and H functions to get F optimization function. G function denotes the movement cost function from the starting node. H function is the so-called heuristic function, which describes the distance of each node to the goal node. (In case of a grid, we should take into account the restricted nodes as well, such as buildings, cranes, material supplies, etc.). H can be defined in many ways: the simplest form of the heuristic function is the sum of horizontal and vertical movements, which is called Manhattan distance. If diagonal movements are allowed in the map, then Chebysev distance is used. If units can move at any angle, then Euclidean distance should be used (Robotin et al., 2010). If the positions of the starting and goal nodes do not depend on time (static case), the heuristic functions of each node are constant values. First, the algorithm creates two lists, the open and the closed one. The closed list contains the nodes which have already been examined; it includes only the starting node at first. The open list contains the parent nodes of the nodes, which are in the closed list (except the nodes which were parents of some nodes from the closed list, but themselves are already in the closed list). The points adjacent to the examined point are called parents. A* calculates the F value for all the parenting nodes of the actual one, and we go in the direction of the smallest F (if there are more than one nodes with equal value, we need to examine all the directions). This node will be part of the closed list, and we continue our computation of its parenting nodes. When there are more than one nodes in the closed list, there will be nodes that share a common parenting node from the open list. This time it has to be checked, whether the recorded F value of this parenting node is smaller than the F value calculated from the actual node and then take the smaller one.

In sensor-based robot motion planning (when obtaining even a feasible solution is very hard, not to mention an optimal one), the planning algorithm may have to run repeatedly, when the sensors detect that the path cannot be crossed. Using A* is inefficient in this case because the moving costs did not change, so the most widely applied algorithm is the Focused Dynamic A* (or D*). Because of the complexity of D*, LPA* (Lifelong Planning A*) has been developed, which is suitable for partially observable graphs. The use of D* Lite (the extended version of LPA*) is proposed by Koenig and Likhachev (2005), when the goal of the search changes. D* Lite determines the same path as D*, but it has a different algorithm and it is shorter, so it is more efficient (Table 1).

Table 1 Comparison between running times of A* algorithms (Robotin et al., 2010)

Dimension	Fast A*	D*	Focused D*
Planning 10^4 cells	5.7 s	8.0 s	6.2 s
Re-planning 10^4 cells	3.0 s	2.1 s	1.3 s
Planning 10^6 cells	37.9 s	55.8 s	50.7 s
Re-planning 10^6 cells	28.2 s	10.1 s	7.6 s
Planning 10^8 cells	136.4 s	335.0 s	298.7 s
Re-planning 10^8 cells	126.8 s	87.4 s	54.3 s

There are two algorithms in graph analysis theory which can be used for pathfinding too. They are the BFS (Breadth First Search) (Vacariu et al., 2007) and the DFS (Depth First Search) (El-Ghoul et al., 2008). Without adapting some other algorithms, the BFS will not be able to find the shortest path, because it is only a strategy for searching a graph, except when the graph is unweighted.

If the graph is unweighted, the shortest path from the starting node is the path until the first visitation of the goal node. The DFS is a similar method for graph analysis, but it does not expand all the nodes of the graph, it just goes as deep as necessary within a child node until the goal is reached. The DFS is primarily used for making a spanning tree of a graph, it always finds a path if one exists (adequate for mazes), but it will not be necessarily the shortest (optimal) one. BFS is complete, but it requires more space and time compared to DFS. They can outperform A* in cases of very small graphs.

Even path planning methods based on randomization were developed and applied in motion planning of robots.

For example: randomized potential field (Barraquand and Latombe, 1991) algorithms (the potential field algorithms use the attraction of the goal and the push of the obstacles, and their mistake is that they might find local minima) and sampling-based algorithms as probabilistic roadmap (PRM) (Amato and Yan, 1996) (for multi-query planning). The most frequently used algorithm (for single-query planning) is the RRT (Rapidly-explored Random Trees) because of its efficiency (LaValle, 1998).

The essence of the RRT is the space filling tree constructed incrementally from randomly drawn samples and growing towards unknown areas. Later researches on RRT have resulted in the RRT connect (trees grow from the starting point and the goal point as well) (Kuffner and LaValle, 2000) and recent ameliorations have brought about the synchronized-biased-greedy RRT, which combines the advantages of the biased version (the tree grows toward to the goal node) and the greedy version (the trees traverse the environment in a single step of the iteration, there is no backtrack) (Yang, 2011). RRTs could not be used for optimal path planning; they found only feasible paths. However, the RRT* algorithm (Fig. 3) was proposed in (Karaman and Frazzoli, 2011) introducing path cost for optimal

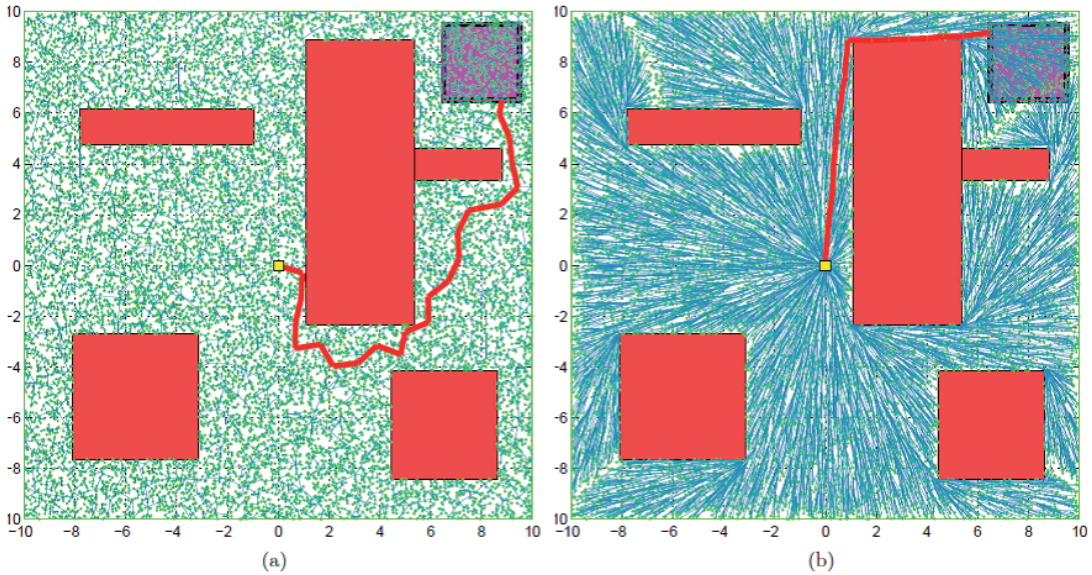

Fig. 3 A Comparison of the RRT (a) and RRT* (b) algorithms on an example (Karaman and Frazzoli, 2011)

motion planning. RRT* is probabilistically optimal (complete). It can iteratively refine its path. Rapidly-Exploring Roadmaps (RRM) (Alterovitz et al., 2011) is an optimal single-query planner as well. It uses the advantages of RRT (for exploration) and applies PRM-type connections along the found paths. Applying RRM is a good trade-off between refinement and exploration, as it can also refine its path iteratively, but unlike RRT, it does not necessarily reach the optimal one. It finds an optimal path as prescribed by the user specified parameter (so it needs to be defined well), before going on with the exploration.

3 Specialties of path planning at a construction site

If we deal with complete information planning problems, the optimal path from different points of view can be chosen by one of the algorithms in Chapter 2 weighting the paths by the importance of each aspect. There are algorithms not mentioned in Chapter 2, which were developed for pathfinding in road networks. We can find similarities between road networks and construction sites, but the main ideas of these algorithms are based on the static feature of road networks. Construction sites, however, cannot be considered static. The first and most important specialty of constructions is that their layout is a time function (dynamic environment). Layout changes are more complex than in the case of road traffic, which is typically shown in the case of reversible direction change. (Bede et al., 2010)

Another property of the construction site motion planning is the size of the construction machinery. While robots can be considered as points (or one element of the grid roadmap), in many cases it is the opposite at constructions, and the size of construction machines have to be computed in most of the cases. In the publication of Lozano-Pérez and Wesley (1979), the Visibility graph approach was presented, where the problem was transformed to pathfinding through the so-called visibility graph VG (L, N). (L is the set of all links, and a straight line connects the i^{th} element of N – node set – to the j^{th} element, and these lines do not overlap any obstacles.) Later, Lozano-Pérez published another approach in (Lozano-Pérez, 1983), called Configuration space (C space) approach, which is the most widely applied one. The coordinates of the configuration space are the degrees of freedom of an object. This way the position and orientation of the object can be represented by one point and the obstacles can be represented by regions (forbidden ranges of coordinates corresponding to DoFs) in C space. Another possibility for considering the machine sizes is shown in (Yu et al., 2013), where a Voronoi diagram is built using the frothing construction algorithm and then topology paths were widened (level-set algorithm was used for the determination of the shortest path in this article). Because someone, who wants to calculate with this factor, can find appropriate possibilities for different optimization methods, we do not deal with this specialty as important decision criteria.

The third property of construction sites is the number of machines working there. In ordinary path planning, we often need to design a path for one robot, while at constructions we always have to consider simultaneously moving machines during the motion of machine for which the path was planned, even if they do not require path planning themselves. An appropriate methodology was proposed in (Tserng et al., 2000), which computes the shortest path for more than one equipment at the same time. Fundamentally, two approaches are used in multiple robot motion planning; these are decoupled and centralized planning. The decoupled planning method has two phases: first it generates collision-free paths for the robots considering only the obstacles, and in the second phase, their relative velocities along the path are determined in order to avoid collision with each other. The centralized method deals with multiple robots in the C-space as if they were parts of one single robot. It is applicable mainly for robot arms instead of mobile robots, and it results in high-dimension spaces. The latest approach in this field is

the consideration of the multiple-robot as a multi-agent system; it is a highly investigated topic in the area of AI of computer games too. Within this topic, we are interested in the so-called cooperative pathfinding problem, which means the agents have full knowledge about other agents and their planned routes. Based on D* (Astengo-Noguez et al., 2010) improved a method using A* for path finding with reservation table (space-time A*) developed by (Silver, 2005). The reservation table has one more dimension than the space where we are planning the path, which is time. It occupies regions where an agent stays in a given time moment. The new algorithm can be applied in 3D, so the reservation table would be 4D, and it considers other dynamic elements as well. Visibility index is assigned to each agent, which says how far ahead they can detect potential collisions, and priority queue is established based on this.

Last but not least, the optimal path is not always the shortest path at construction sites, as multi-criteria optimization is necessary. In most of the cases, the shortest path algorithms are capable of considering the weighting of different path sections (for example by weighting graph's edges). In this case, one of the criteria of the search can be the short path, and other issues (safety, visibility) can be implemented immediately as other criteria. For the solution, e.g. genetic algorithm (GA) is used, which is a random search method that mimics the process of natural selection, and genetics are applied in (AL-Taharwa et al., 2008). It starts with candidate solutions coded in binary strings with a set of properties and an objective function, which evaluates their performance as a problem solution. Applying reproduction, mutation, and crossover, the GA generates the new population of solutions. Soltani and Fernando (2004) considered safety and visibility too with the help of fuzzy sets.

4 Evaluation of path planning algorithms

Some pathfinding algorithms described in Chapter 3 were excluded from our decision process on method selection because of their properties mentioned before. The candidate algorithms from which we want to select the best for construction site path planning are the followings: Dijkstra, A*, RRT*, GA.

As written in Chapter 3, construction sites' transport processes have special features, which must be handled by the selected method in order to get realistic results. These are: dynamic environment and traffic optimization for multiple vehicles at the same time, and also other criteria has to be considered besides distance only for optimization.

Compared with the other algorithms, Dijkstra proves to be the best if we want to search for a path from one source to multiple destinations. A* is adequate in many cases if high-dimensions are ignored. RRT* fits the most high-dimensional configuration spaces, because it is probabilistically complete. GA can be the most efficient if numerous criteria optimization is required, in that case, other algorithms need the supplement. In Table 2, the algorithms can be seen as a function of our criteria.

Table 2 Comparison of path planning algorithms

Algorithms/ Applicability to the condition	Dynamic environment	Multiple machines	Multi-criteria optimization
Dijkstra	satisfactory	suitable with addition	suits with predefined (fuzzy sets) path costs
A*	adequate	perfectly suitable (space-time A*)	suits with predefined (fuzzy sets) path costs
RRT*	perfectly suitable (for high-dimensions)	suitable with addition	suits with predefined (fuzzy sets) path costs
GA	suitable	suitable with addition	perfectly suitable

$T = O$ (*No. of interm.points · Population size · No. of generations*)

The algorithm's applicability to a dynamic environment (how fast the algorithm can generate a new path) is measured in our deliberation as the algorithm execution time, which can be independent of the computer, if we use the complexity function (T), which characterizes the steps taken by the algorithm as a function of length of the string representing the input size (n).

In case of Dijkstra and A* $T(n) = 2(n-1)^2$ so the asymptotic time complexity (leaving the coefficients and lower order terms) $O(n^2)$. In the case of GA used in (Soltani et al., 2002)

Reading the measured data from the article, A* and Dijsktra had the same execution time (7 s, 120 s) for the same grid size (40, 80), and GA had a lower (5 s, 51 s) one. RRT* time complexity is $O(n)$ for the number of samples n in fixed environment.

5 The proposed intelligent path planning method

In the proposed method, we represent the construction site with a grid network because of the advantages of this structure mentioned earlier. In our approach, the construction site is a predictable dynamic environment. In spite of its continuous changes, we can predict the significant changes from the schedule and we assume that we know the path of all other moving objects. Sensors are operated for safety reasons only. That is why global pathfinding algorithms can be applied too. The algorithm has to run few times, only in the case of progress in the schedule (or if sensors detect deviation from the prescribed environmental data). Using the results of the algorithm evaluation of the previous chapter, we will apply the A* method for path finding, and because the construction site with its equipment is assumed to be a multi-agent system, an algorithm using the basics of space-time A* (3D representation with time intervals) and decoupled planning method (the velocity of machines

vary) will be used. The system establishes priority among the equipment, taking into account their actual lateness compared to the scheduling. On the basis of this priority, the shortest paths will be determined so that the equipment with the lower priority has to wait if it wants to travel through the same area (cell), while this area is occupied by a higher priority machine. The cells are occupied for specific time intervals necessary for the given machine with its actual velocity to get through this area. Waiting can mean slowing down (so the higher priority equipment can leave the area until the slowed down one arrives), or stopping (the lower priority vehicle will stop until the other one has left the problematic area. This can be the case, for example, when the equipment with the higher priority blocks the only passable route).

In order to get results optimized for multi-criteria, we determine 'imaginary distances' between cells instead of real distances, which will be the input data of the space-time A*. These imaginary distances are fuzzy-weighted safety factors added to real distances forming the travel cost between cells. The safety issues to be considered are the following: First, the distance of the equipment from an obstacle; the farther the machine goes from an object, the higher the possibility is to evade a collision (or avoid the danger e.g. load dropping from a crane). The second factor is arriving at 'hazardous areas'. The construction site cannot be imagined as a green area, where all the unoccupied areas can be travelled unconditionally. In this viewpoint, it is rather similar to a road network with safe, trodden paths, from which deviation is possible. So we assign a risk value proportional to the hazard of travelling to all the areas (for example a high hazard area for a vehicle to travel through – a risk, which does not derive from the presence of an obstacle – is where manual labour is done by human power).

From these two safety factors, 'safety distances' (costs) will be calculated with the help of fuzzy rules, then the cost will be added to the real distance. We consider the obstacles closer than three cells to the equipment for which the path planning is calculated, and objects farther than this are assumed to be harmless. The applied fuzzy if-then rules are the following:

1. If the obstacle is close then the cost is high.
2. If the obstacle is in the moderate distance and risk of the area is hazardous then the cost is moderate.
3. If the obstacle is far and risk of the area is non-hazardous then the cost is low.

The input fuzzy sets can be seen in Fig. 4. The minimum distance to an obstacle is one cell. The output fuzzy data is a weighted cost (safety distance) determined by the Takagi-Sugeno fuzzy method. If we add this to the real distances, we get the imaginary distances. The input set of area hazard is without dimension, it comes from the expert's judgment.

The flowchart of the method can be observed in Fig. 5. First, the allocation of the input parameters takes place from

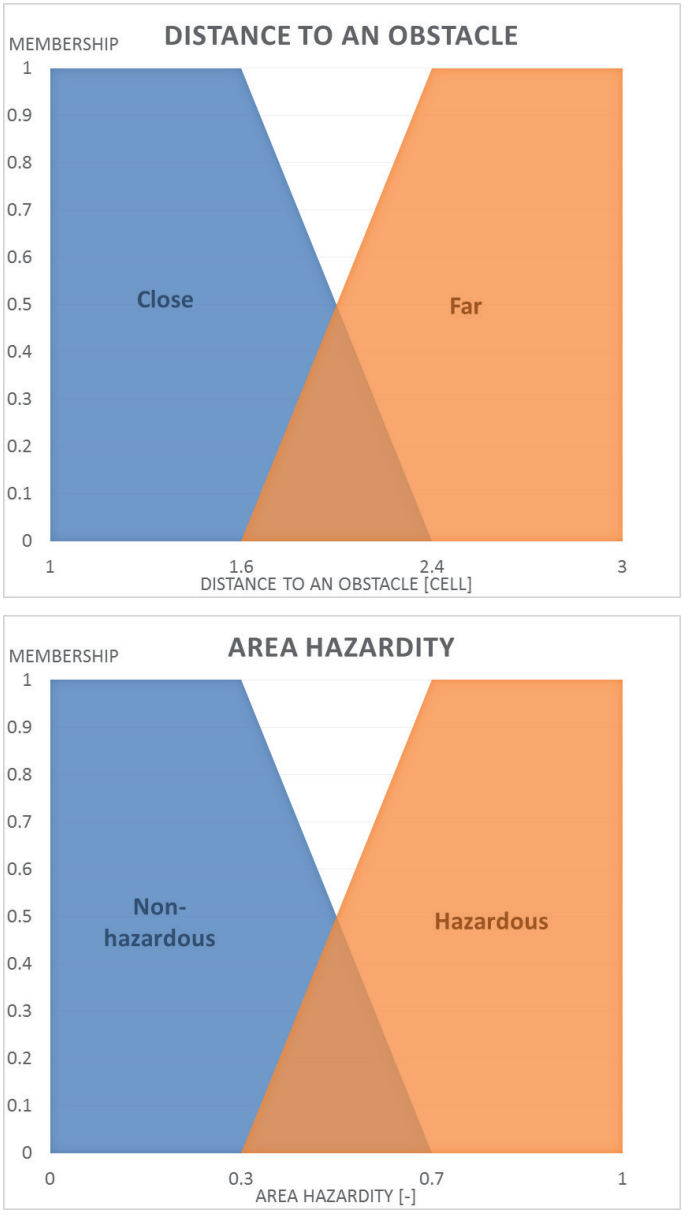

Fig. 4 Input fuzzy sets of calculating safety distances

the construction layout, then path and velocity of all the vehicles in priority order are calculated ("…" means towards the lower priority vehicles). If a collision would happen, there are two possibilities to avoid that: either by speed reduction of the vehicle with the lower priority, or (if that is not possible, because a higher priority vehicle occupies its target for a long time period) generating a new path for it.

6 Evaluation of the proposed method

For illustration and explanation purposes, a simple example is presented, processed by the proposed path planning method.

This particular project is a road construction, where the concrete road crosses a natural obstacle, for example, a river (a bridge will solve the traversability). In this case, the river and the road being constructed highly restrict the traversable areas for the trucks, so crossing each other's paths during their operation has a high possibility. Three pieces of equipment are

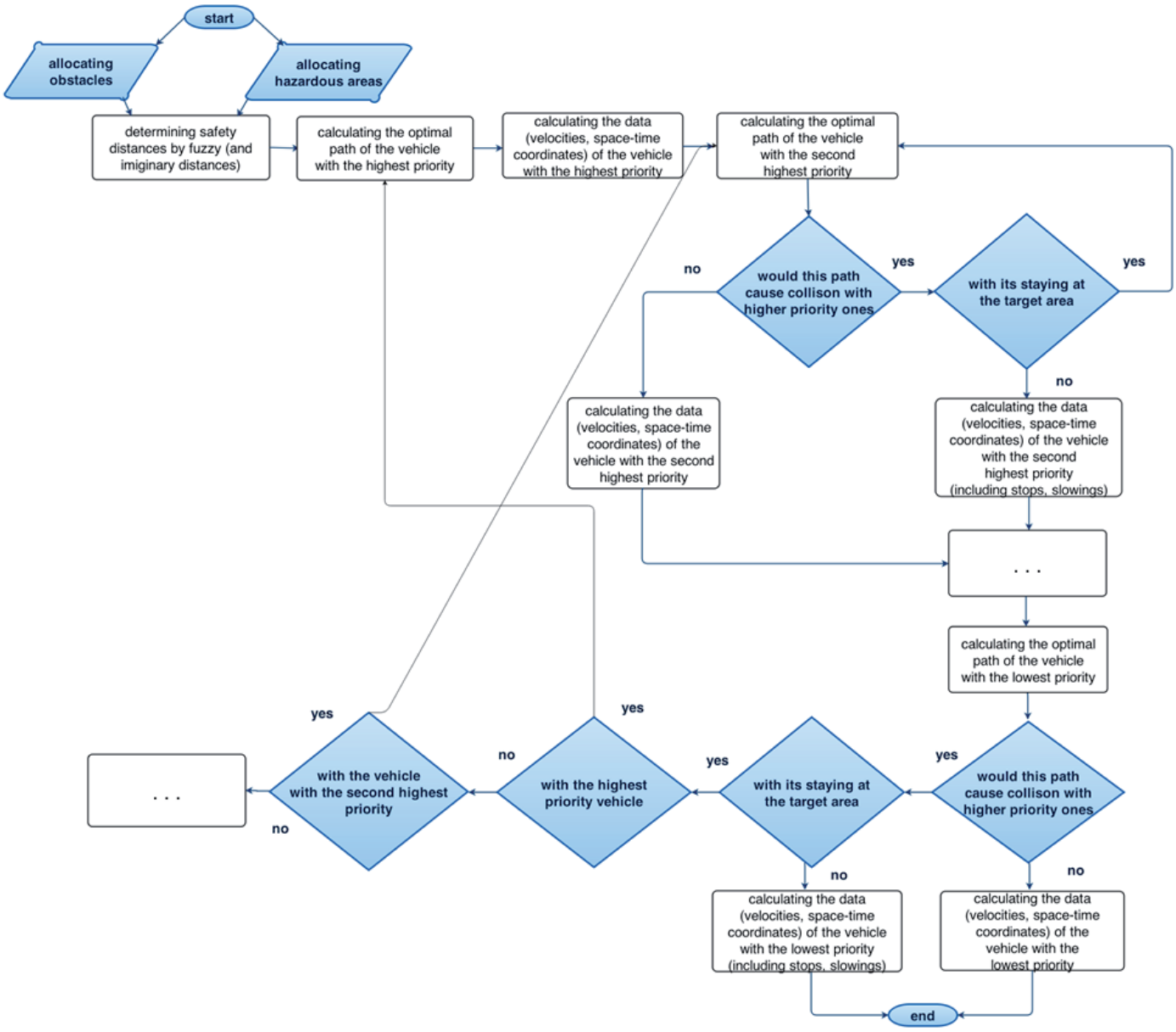

Fig. 5 Flowchart of path planning method

considered in this example. Two earthmoving trucks for landfilling: one is on the road to the target location and the other is leaving the target location on its way back to the excavation area. The third truck is transporting concrete from the concrete plant. The 'layout' of the construction site with the starting and the end positions of the vehicles can be seen in Fig. 6. Hazardous areas were not considered in this basic example (or all the areas were considered as areas with 0 % risk), but 'imaginary' and 'safety' distances are calculated because an obstacle is presented. (The algorithm assigns each cell an additional traveling cost inversely proportional to the distance of the cell and the closest obstacle to it.)

We divided the construction site with the help of a 10x10 grid into sub-regions. The start and target locations are indicated in Fig. 6 and in Table 3 as well.

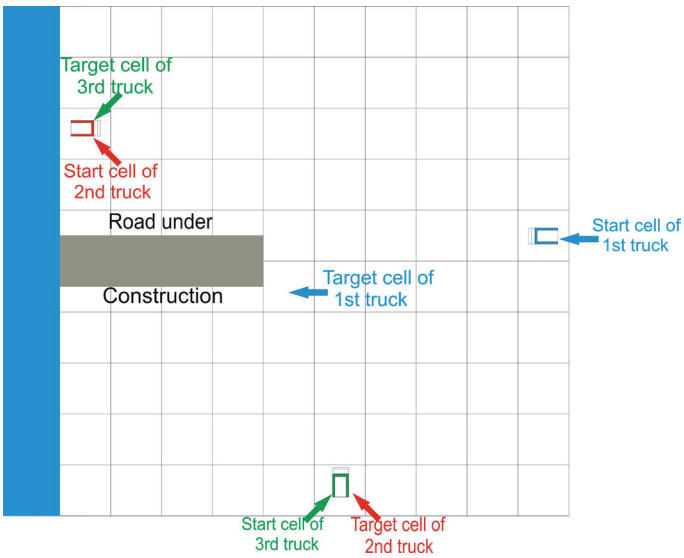

Fig. 6 Outline of the example

Table 3 Starting and end point of vehicles in the example

Coordinates/Vehicles	Start X	Start Y	Target X	Target Y
1st truck	10	6	5	5
2nd truck	1	8	6	1
3rd truck	6	1	1	8

For the sake of simplicity, we assume the constant velocity for all three vehicles (however, the method would allow different values as well, and it would not require any additional programming) with a unit value. We set the algorithm so that the vehicle which should have waited decreases its velocity to 40 % of the previous speed two cells before the roads are crossing each other (or to 16 % if there is only one cell to the crossing cell, or stop – until it is necessary – if the crossing cell is right next to the starting cell. The starting cell could not be a crossing cell because one vehicle must have left it by the time the other one arrives), but it can be also changed as an input parameter of the algorithm. Resulting data of the example is shown by Table 4.

We can see in the example, the algorithm considers the priorities and it works properly. For a better understanding of the algorithm, the results are illustrated in Fig. 7, which shows some of the working principles of the algorithm. The 1st truck had the highest priority, so its optimal path was computed first, as a function of time. Then the optimal path of the 3rd truck with the second priority (in Fig. 7 numbering was done by priority order) was computed. In order to avoid the collision, its velocity was decreased the above-mentioned way. During the computation of the optimal path of the 2nd truck (lowest priority), the algorithm allows it to follow its optimal path (the path of the 3rd truck in reverse order) except for bypassing the cell, where the collision (which cannot be avoided by slowing down or waiting) would happen. The bypass can happen only in the indicated (with a red line) direction because of two reasons: first, it is the safer one, and second, because in the other direction it would arrive at the target area of the first truck at the same time interval when the first truck has arrived (and stays there).

7 Conclusion

Above tests have proven that the developed algorithm is capable of coping with situations of multi-vehicle traffic in construction sites. Parking on frequently used areas and handling bottlenecks pose no challenge and do not lead to deadlock situations. Therefore, the method was decided to be integrated into a complex logistics model, aimed by our research. Next steps will be the examination of the method with respect to integration into navigation systems of real machinery and into the material flow simulation models.

Table 4 Example output dat

1st truck (highest priority)

Cell X coordinate	Cell Y coordinate	Velocity through the cell	Cell occupying start time	Cell occupying end time
10	6	1	0	1.41
9	5	1	1.41	2.41
8	5	1	2.41	3.41
7	5	1	3.41	4.41
6	5	1	4.41	5.41
5	5	1	5.41	Inf

please note: Inf means the truck occupies the cell to the next running period (the vehicle stays here while it dumps or during its loading).

3rd truck (second highest priority)

Cell X coordinate	Cell Y coordinate	Velocity through the cell	Cell occupying start time	Cell occupying end time
6	1	1	0	1
6	2	1	1	2
6	3	0.4	2	4.5
6	4	0.4	4.5	7
6	5	1	7	8
6	6	1	8	9.41
5	7	1	9.41	10.83
4	8	1	10.83	11.83
3	8	1	11.83	12.83
2	8	1	12.83	13.83
1	8	1	13.83	Inf

2nd truck (lowest priority)

Cell X coordinate	Cell Y coordinate	Velocity through the cell	Cell occupying start time	Cell occupying end time
1	8	1	0	1
2	8	1	1	2
3	8	1	2	3
4	8	1	3	4.41
5	7	1	4.41	5.83
6	6	1	5.83	7.24
7	5	1	7.24	8.66
6	4	1	8.66	9.66
6	3	1	9.66	10.66
6	2	1	10.66	11.66
6	1	1	11.66	Inf

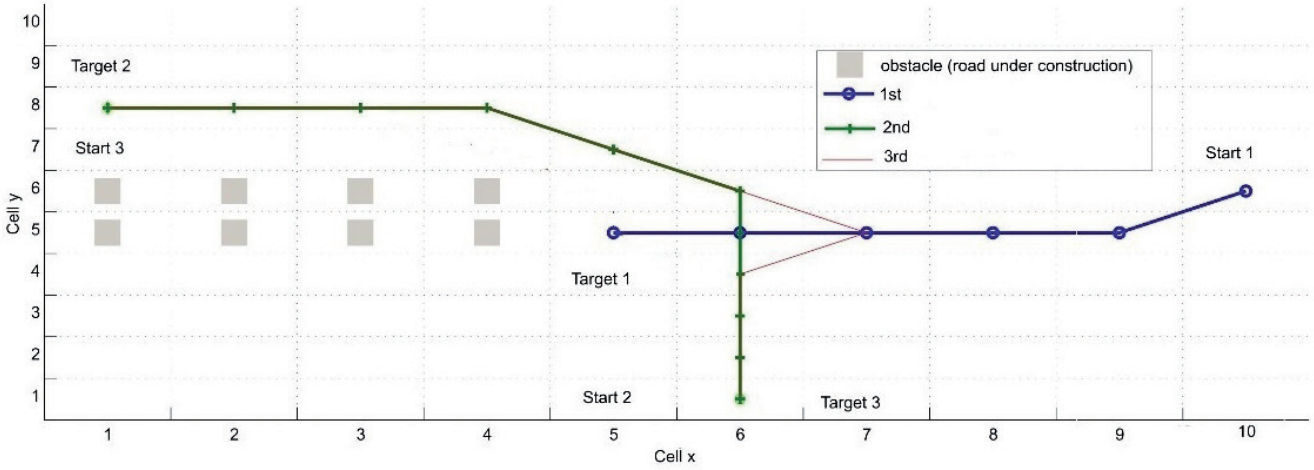

Fig. 7 Determined of optimal path of the example

Acknowledgement

This paper is a part of our research project (KTIAAIK-12-1-2013-0009) financed by the National Development Agency of Hungary, total financial support is HUF 419 904 851) which aims to improve logistics processes in the building industry.

References

AL-Taharwa, I., Sheta, A., Al-Weshah, M. (2008) A mobile robot path planning using genetic algorithm in static environment. *Journal of Computer Science*, 4(4), pp. 341-344. DOI: 10.3844/jcssp.2008.341.344

Alterovitz, R., Patil, S., Derbakova, A. (2011) Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning. In: *2011 IEEE International Conference on Robotics and Automation*, pp. 3706-3712. DOI: 10.1109/ICRA.2011.5980286

Amato, N. M., Wu, J. (1996) A randomized roadmap method for path and manipulation planning. In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, Vol. 1, pp. 113-120. DOI: 10.1109/ROBOT.1996.503582

Anastopoulos, N., Nikas, K., Goumas, G., Koziris, N. (2009) Early experiences on accelerating Dijkstra's algorithm using transactional memory. In: *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp.1-8, 23-29 May 2009. DOI: 10.1109/IPDPS.2009.5161103

Andrejiova, M., Grincova, A., Marasova, D., Fedorko, G., Molnar, V. (2014) Using logistic regression in tracing the significance of rubber–textile conveyor belt damage. *Wear*, 318(1-2), pp. 145-152. DOI: 10.1016/j.wear.2014.06.026

Astengo-Noguez, C., Sanchez-Ante, G., Calzada, J. R., Hernandez, S. (2010) Multi-Robot Collective Path Finding in Dynamic Environments. In: Barrera, A. (ed.) *Mobile Robots Navigation*, InTech, pp. 307-329. DOI: 10.5772/8983

Barraquand, J., Latombe, J.-C. (1991) Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research,*10(6), pp. 628-649. DOI: 10.1177/027836499101000604

Bede, Z., Szabó, G., Péter, T. (2010) Optimalization of road traffic with the applied of reversible direction lanes. *Periodica Polytechnica Transportation Engineering*, 38(1), pp. 3-8. DOI: 10.3311/pp.tr.2010-1.01

Bellman, R. (1958) On a routing problem. *Quarterly of Applied Mathematics*, 16, pp. 87-90.

van den Berg, J. (2007) *Path planning in dynamic environments*, PhD Thesis. Drukkerij Bariet: Ruinen.

Burlet, J., Aycard, O., Fraichard, T. (2004) Robust motion planning using Markov decision processes and quadtree decomposition. In: *Robotics and Automation, 2004. Proceedings. ICRA ,04. 2004 IEEE International Conference on*, Vol. 3, pp. 2820-2825, 26 April-1 May 2004. DOI: 10.1109/ROBOT.2004.1307488

Dijkstra, E. W. (1959) A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), pp. 269-271. DOI: 10.1007/BF01386390

El-Ghoul, S., Hussein, A. S., Wahab, M. S. A. (2008) A modified multiple depth first search algorithm for grid mapping using mini-robots Khepera. *Journal of Computing Science and Engineering*, 2(4), pp. 321-338.

Galceran, E., Carreras, M. (2013) A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12), pp. 1258-1276. DOI: 10.1016/j.robot.2013.09.004

Hart, P. E., Nilsson, N. J., Bertram, R. (1972) Correction to 'A formal basis for the heuristic determination of minimum cost paths'. *ACM SIGART Bulletin*, 37, pp. 28-29. DOI: 10.1145/1056777.1056779

Hart, P. E., Nilsson, N. J., Raphael, B. (1968) A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), pp. 100-107. DOI: 10.1109/tssc.1968.300136

Johnson, D. B. (1977) Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1), pp. 1-13. DOI: 10.1145/321992.321993

Karaman, S., Frazzoli, E. (2011) Sampling-based Algorithms for Optimal Motion Planning. *The International Journal of Robotics Research*, 30(7), pp. 846-894. DOI: 10.1177/0278364911406761

Koenig, S., Likhachev, M. (2005) Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3), pp. 354-363. DOI: 10.1109/TRO.2004.838026

Kuffner, J. J., LaValle, S. M. (2000) RRT-connect: An efficient approach to single-query path planning. In: *Robotics and Automation, 2000. Proceedings. ICRA ,00. IEEE International Conference on*, Vol. 2, pp. 995-1001. DOI: 10.1109/ROBOT.2000.844730

Kunigahalli, R., Russell, J. S., Skibniewski, M. J. (1994) Motion planning for automated construction. *Automation in Construction*, 3(1), pp. 71-78. DOI: 10.1016/0926-5805(94)90034-5

LaValle, S. M. (1998) Rapidly-Exploring Random Trees: A new tool for path planning, *Technical Report* No. 98-11,: Department of Computer Science, Iowa State University.

Lozano-Pérez, T., Wesley, M. A. (1979) An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10). pp. 560-570. DOI: 10.1145/359156.359164

Lozano-Pérez, T. (1983) Spatial planning: a configuration space approach. *IEEE Transactions on Computers*, C-32(2), pp. 108-120. DOI: 10.1109/TC.1983.1676196

Robotin, R., Lazea, G., Marcu, C. (2010) Graph search techniques for mobile robot path planning. In: Dudas, L. (ed.) *Engineering the Future*, InTech, pp. 159-178, DOI: 10.5772/10373

Sharma K. R. (2013) Design and implementation of path planning algorithm for wheeled mobile robot in a known dynamic environment. *IJRET: International Journal of Research Engineering and Technology*, 2(06), pp. 967-970.

Silver, D. (2005) Cooperative Pathfinding. In: Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, June1–3, 2005, Marina del Rey, California. The AAAI Press, Menlo Park, California, pp. 117-122.

Solka J. L., Perry, J. C., Poellinger, B. R., Rogers, G. W. (1995) Fast computation of optimal paths using a parallel Dijkstra algorithm with embedded constraints. *Neurocomputing*, 8(2), pp. 195-212. DOI: 10.1016/0925-2312(94)00018-N

Soltani, A., Fernando, T. (2004) A fuzzy based multi-objective path planning of construction sites. *Automation in Construction*, 13(6), pp. 717-734. DOI: 10.1016/j.autcon.2004.04.012

Soltani, A. R., Tawfik, H., Goulermas, J. Y., Fernando, T. (2002) Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms. *Advanced Engineering Informatics*, 16(4). pp. 291-303. DOI: 10.1016/S1474-0346(03)00018-1

Stehlíková, B., Molnár, V., Fedorko, G. (2014) Possibilities of Experiments and of Using Experimental Results Obtained from the Test Equipment for Measuring Properties of Conveyor Belts Pipe Conveyor. *Applied Mechanics and Materials*, 683, pp. 165-170. DOI: 10.4028/www.scientific.net/AMM.683.165

Tserng, H. P., Ran, B., Russell, J. S. (2000) Interactive path planning for multi-equipment landfill operations. *Automation in Construction*, 10(1), pp. 155-168. DOI: 10.1016/S0926-5805(00)00073-X

Vacariu, L., Roman, F., Timar, M., Stanciu, T., Banabic, R., Cret, O. (2007) *Mobile robot path planning implementation in software and hardware*. In: Long, C. A., Mladenov, V. M., Bojkovic, Z. (ed.) ISPRA'07 Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation, WSEAS, Stevens Point, USA, pp. 140-145.

Yang, K. (2011) Anytime synchronized-biased-greedy rapidly-exploring random tree path planning in two dimensional complex environments. *International Journal of Control, Automation and Systems,* 9(4), pp. 750-758. DOI: 10.1007/s12555-011-0417-7

Yu, C., Qiu, Q., Chen, X. (2013) A hybrid two-dimensional path planning model based on frothing algorithm construction algorithm and local fast marching method. *Computers and Electrical Engineering*, 39 (2), pp. 475-487. DOI: 10.1016/j.compeleceng.2012.09.010